Propuesta de trabajo Final de la Carrera Ingeniería de Sistemas

UN ENFOQUE BASADO EN QUERIES PARA LA RECONSTRUCCIÓN DE ARQUITECTURAS

Director: Ing. Álvaro Soria

Co-Director: Dr. J. Andrés Diaz Pace

Alumnos: Nicolás Frontini y Facundo Maldonado

1. Introducción

La Arquitectura de Software de un sistema sirve para capturar las principales decisiones de diseño del sistema. Estas primeras decisiones son las más difíciles de tomar correctamente, las más difíciles de cambiar más adelante en el proceso de desarrollo y sus efectos son los que más perduran en el ciclo de vida de un sistema. En particular, una arquitectura abarca aspectos estructurales y de comportamiento: los aspectos estructurales se refieren a vistas estáticas del sistema, mientras que los aspectos de comportamiento se refieren a vistas dinámicas del sistema. Por ejemplo, los diagramas de componentes sirven para representar aspectos estructurales, mientras que los Use Case Maps [4] proveen un modelo para aspectos de comportamiento. Así y todo, aún con la ayuda de las arquitecturas, entender cómo funciona un sistema complejo y comunicárselo a los demás continua siendo un problema para muchas organizaciones.

En general, todo sistema que involucra software posee una arquitectura de software. Sin embargo, no necesariamente existe en la práctica una documentación acertada de esta arquitectura. Suele suceder que la gente que diseñó el sistema se ha ido de la organización, que la documentación es escasa y está desactualizada, que el código fuente se ha perdido, y lo único con que se cuenta es con el código ejecutable binario. En estas situaciones, dado que una arquitectura puede existir independientemente de su especificación, es que cobran importancia las enfoques de documentación y reconstrucción de arquitecturas[2].

2. Motivación

La Reconstrucción de Arquitecturas es una técnica para generar representaciones arquitectónicas de un sistema que puede ser utilizada de diversas formas. El principal uso de esta representación es el de documentar la arquitectura existente de un sistema. Si no existe la documentación o la documentación disponible esta desactualizada, la representación obtenida mediante la reconstrucción puede ser usada como una base para la re-documentación de la arquitectura. Esta representación también puede ser usada como un punto de comienzo para realizar re-ingeniería sobre el sistema y crear una nueva arquitectura. Finalmente, la representación puede ser usada como una forma de identificar componentes reusables ó para establecer una arquitectura base dentro de una línea de productos de software [10].

Algunas soluciones a la Reconstrucción de Arquitecturas han sido propuestas en [3, 7, 12, 9, 10, 8]. En la mayoría, la arquitectura del sistema se extrae principalmente por medio del análisis del código fuente. En otras, la información se obtiene tanto del código fuente como del sistema en ejecución ó sólo reconstruye arquitecturas de sistemas implementados utilizando patrones de diseño. Muchos de los enfoques mencionados sólo muestran aspectos estructurales (diagramas de clases) y muy pocos muestran resultados que permitan visualizar el comportamiento del sistema (ej: diagramas de UCMs ó de secuencia).

Estos enfoques presentan el resultado de la reconstrucción en un solo paso y el arquitecto no ve el progreso del proceso de reconstrucción. Esto hace que el proceso no se pueda "guiar" a medida que se reconstruye la arquitectura. Es decir, no se tiene control sobre qué y cómo se recupera la información, ni puede intervenir si lo considera necesario. Además, no es posible manipular la información de forma interactiva,

perdiendo así la posibilidad de contar con la experiencia del arquitecto y su conocimiento en el dominio. Por otro lado, ninguno de estos enfoques mantiene la información recuperada de tal forma que sea flexible su manipulación.

3. Propuesta

La propuesta de este trabajo consiste en recuperar la arquitectura de un sistema ya implementado. Con el objetivo de superar las limitaciones de los enfoques mencionados anteriormente, se pretende recuperar tanto la estructura como el comportamiento del sistema a analizar.

A partir de la información estática se pretende extraer los componentes arquitectónicos. Como lenguaje de especificación estructural para esto se sugieren los diagramas de componentes UML.

Además del modelo estructural, se debería soportar un modelo de comportamiento. Debido a que el código fuente no aporta toda la información relevante del comportamiento del sistema, la información obtenida del sistema en tiempo de ejecución podría contribuir al modelo de comportamiento. Para dicho modelo se sugieren los diagramas de UCMs.

Otro de los puntos de este trabajo, planteado principalmente para atacar la mayor deficiencia de los enfoques actuales, seria el permitir la interacción del usuario (arquitecto) en el proceso de reconstrucción. El arquitecto participaría refinando la especificación. De esta manera, se le otorgaría la posibilidad de visualizar (por medio de los diagramas) los resultados intermedios y se le permitiría incrementar a cada paso el nivel de detalle.

La información extraída del sistema debería estar representada de forma tal que permita ser manipulada de forma sencilla y flexible. Teniendo en cuenta que la programación lógica permite realizar consultas flexibles, la información podría estar representada con hechos y reglas al estilo Prolog, para lo cuál se podría utilizar JavLog [1] ó JQuery [13, 6].

La Figura 1 muestra un esquema del enfoque a desarrollar. La información estática es procesada por la herramienta (Tool). Luego, se generan vistas parciales que son validadas por el arquitecto. En base a lo observado, el arquitecto puede interactuar con la herramienta para refinar aún más la especificación y volver a obtener nuevas vistas.

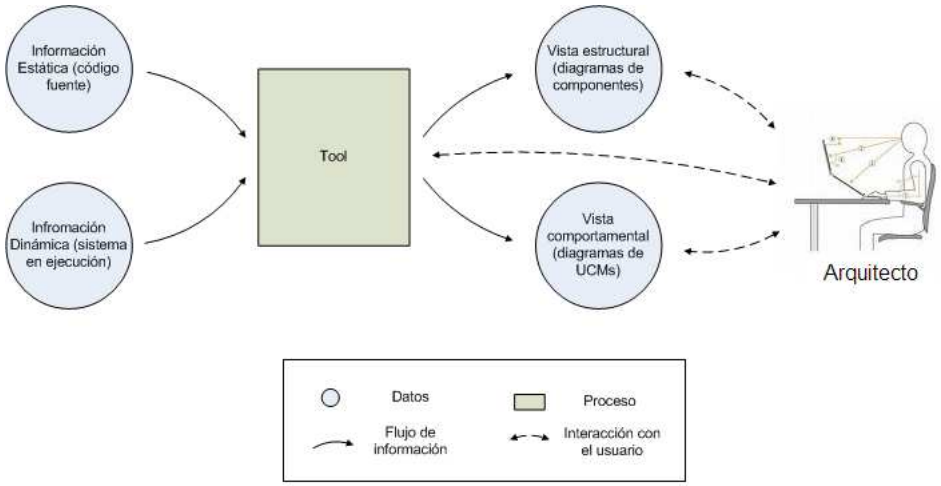

Figura 1: Propuesta de Recuperación de Arquitecturas.

Para la implementación del trabajo se tomará como base el proyecto FLABot [11], ya que posee un editor de Componentes, un editor de Use Case Maps, ambos integrados a la plataforma Eclipse[5].

Finalmente, una contribución del trabajo es facilitar la tarea de los mismos arquitectos del sistema: ya que al tener vistas que representan la estructura y el comportamiento de un sistema, se pueden detectar con mayor claridad violaciones a condiciones estructurales y/o comportamentales cometidas durante el desarrollo o el mantenimiento.

4. Cronograma de actividades

A continuación se presenta el cronograma de las actividades propuestas –y su duración estimada en meses– que conformarán el desarrollo de nuestro trabajo final. El tiempo total del trabajo se fijó en aproximadamente ocho meses.

Actividad	**Duración Estimada**
Relevamiento de Información sobre reconstrucción de Arquitecturas	2 semanas
Evaluación de los enfoques de reconstrucción de Arquitecturas	3 semanas
Planteo de los requerimientos (*)	4 semanas
Análisis y diseño, basado en los requerimientos planteados (*)	5 semanas
Implementación del prototipo (*)	8 semanas
Elaboración de ejemplos y testeo del prototipo (*)	5 semanas
Documentación y elaboración del informe (*)	7 semanas

Nota: la herramienta será diseñada e implementada mendiante una serie de prototipos incrementales. Por lo tanto, algunas de las tareas marcadas con (*) podrán ser realizadas en paralelo.

Referencias

[1] A. Amandi, M. Campo, and A. Zunino. JavaLog: A framework-based integration of Java and Prolog for agent-oriented programming. *Computer Languages, Systems and Structures*, 31, 2004. ISSN 0096-0551.

[2] L. Bass, P. Clement, and R. Kazman. *Software Architecture in Practice. 2ed.* Addison-Wesley, 2003. ISBN 0-321-15495-9.

[3] I. T. Bowman, R. C. Holt, and N. V. Brewster. Linux as a case study: Its extracted software architecture. In *ICSE'99*, Los Angeles, CA, USA, 1999.

[4] R. J. A. Buhr and R. S. Casselman. *Use Case Maps for Object-Oriented Systems*. Prentice Hall, 1996. ISBN 0-13-456542-8.

[5] IBM Corporation and The Eclipse Foundation. Eclipse platform technical overview. http://www.eclipse.org/articles/Whitepaper-Platform-3.1/eclipse-overview-2005-12.pdf.

[6] Kris De Volder Doug Janzen. Navigating and querying code without getting lost. *Department of Computer Science University of British Columbia*, 2003.

[7] H. Gall, R. Klosch, and R. Mittermeir. Object-oriented re-architecting. In *ESEC-5*, Berlin, Germany, 1995.

[8] G. Y. Guo, J. M. Atlee, and R. Kazman. A software architecture reconstruction method. In *WICSA-1*, San Antonio, 1999.

[9] Yeh A. S. Harris D. R., Reubenstein H. B. *Reverse Engineering to the Architectural Level*. ACM Press, 1995.

[10] Chris Verhoef Rick Kazman, Liam OŠBrien. Architecture reconstruction guidelines. Technical report, CMU-SEI, 2003.

[11] Alvaro Soria. Architecture-driven fault localization of implicit invocation systems. In *Proceedings ASSE 2004, dentro de las 33 Jornadas Argentinas de Informática e Investigación Operativa (JAIIO)*, Córdoba, Argentina, September 2004.

[12] V. Tzerpos and R. C. Holt. A hybrid process for recovering software architecture. In *CASCON'96*, Toronto, Canada, November 1996.

[13] Kris De Volder. Jquery: A generic code browser with a declarative configuration language. *University of British Columbia*, 2006.